# The Non-Regularity of an Even-Length Palindrome with Suffix

Zach Tomaszewski
*Department of Information and Computer Sciences*
*University of Hawaii at Manoa*
*Honolulu, HI 96822*
*Email: ztomasze@hawaii.edu*

**Abstract**

The class of regular languages plays an important role in theoretical computer science. It is therefore important to be able to recognize which languages are regular and which are non-regular. This paper examines the regularity of a language based on even-length palindromes, proving it to be non-regular through the application of the well-known pumping lemma. It then shows how the pumping lemma is inadequate for proving the non-regularity of a slightly more complex language based on an even-length palindrome with an arbitrary suffix. Instead, the Myhill-Nerode theorem is necessary to prove that this second language is also non-regular.

**Keywords**

regular language; palindromes; pumping lemma; Myhill-Nerode theorem;

## I. Introduction

The class of regular languages are an important category of simple languages in computer science. Given an alphabet $\Sigma$ of non-divisible symbols, a regular language can be described using set notation:

1) $\emptyset$, $\{\epsilon\}$ and $\{\mathtt{a}\}$ for every $a \in \Sigma$ are regular sets over $\Sigma$.
2) If $X$ and $Y$ are regular sets over $\Sigma$, then $X \cup Y$, $XY$ and $X^*$ are also regular sets over $\Sigma$.
3) X is a regular set over $\Sigma$ only when it can be constructed by applying (1) and (2) a finite number of times [1] [2].

Regular languages can also be defined in terms of regular grammars, regular expressions, and finite automata. For regular languages, these are all essentially equivalent formulations. A regular grammar produces a regular language; a regular expression describes or matches a regular language; a finite automata accepts a regular language as input.

Because the characteristics of regular languages are so well-known, it is often useful to know whether a given language is regular. For example, if a language is shown to be regular, then it necessarily follows that the language could be described using a regular expression or that a finite automata could be constructed to recognize strings from the language.

Similarly, it is often useful to show that a given language is *not* regular. The pumping lemma is often used for this purpose.

The pumping lemma essentially assumes the existence of a finite automaton $M$ that accepts a given regular language $L$. By definition, $M$ has a sufficient number of states to accept any string in $L$. Therefore, if some string $w \in L$ contains more symbols than the number of states in M, then some internal portion of $w$ must have been repeated or "pumped." That is, while accepting $w$, $M$ must have completed a cycle, returning to the same state one or more times.

The pumping lemma is stated more formally (without proof) as follows: If $L$ is a regular language, then there is a number $p$ (the pumping length) such that, if $w$ is any string in $L$ and $|w| \geq p$, then $w$ may be divided into 3 pieces, $w = xyz$, such that $|xy| \leq p$, $|y| \geq 1$, and for all $i \geq 0, xy^iz \in L$ [3] [4] [5].

The pumping lemma has already been used to prove that a number of simple languages are not regular. Such languages include:

- $L = \{\mathtt{0}^{i^2} | i$ is an integer and $i \geq 1\}$ [4]
- $L = \{w | w$ is formed by an equal number of 0s and 1s$\}$ [3]
- $L = \{0^n 1^n | n \geq 0\}$ [3]
- $L = \{ww | w \in \{\mathtt{0}, \mathtt{1}\}^*\}$ [3]

Such proofs usually take the form of an "adversary argument" [4] in which a theoretical adversary is allowed to choose any positive value for $p$. The proof author then chooses an example string $w \in L$ based on $p$ and demonstrates that there is no possible assignment of $w = xyz$ such that $xy^iz \in L$ for some $i$.

It is important to note that the pumping lemma provides only a necessary condition for a language to be regular. That is, if the language is regular, then some part of any sufficiently-long string in the language can be pumped to produce another string in the same language. However, this is not a sufficient condition. Thus, the reverse is not necessarily true. That is, simply finding a language that can be pumped does not mean that it must be a regular language.

This fact also affects pumping lemma proofs that show a particular language is *not* regular. Such proofs show that a sufficiently-long string of a certain form cannot be pumped according to the rules of the pumping lemma. This forces the conclusion that the given language is not regular. However, finding that all long strings in the language are pumpable is simply inconclusive. It does not prove that the language is regular.

The Myhill-Nerode theorem is an alternative to the pumping lemma that provides both necessary and sufficient conditions for a regular language. It also implies that all finite automata that accept a given language can be minimized to the same single form. Like the pumping lemma, the Myhill-Nerode theorem considers how a finite automata processes strings in a given regular language. Assume a finite automata, $M$, capable of accepting two different strings, $xz$ and $yz$, in a language $L$ over $\Sigma$. If, after reading either $x$ or $y$, reading any $z \in \Sigma^*$ leaves the machine in the same resulting state, then $x$ and $y$ from an equivalence class [4] and so are indistinguishable with respect to $L$ [6]. Furthermore, any regular language has a finite number of such equivalence classes [4].

More formally, the Myhill-Nerode theorem (presented here without proof) states that:

The following three statements are equivalent:

1) The set $L \subseteq \Sigma^*$ is accepted by some finite automata.
2) $L$ is the union of some of the equivalence classes of a right invariant equivalence relation of finite index.
3) Let an equivalence relation $R_L$ be defined by: $xR_Ly$ if and only if for all $z$ in $\Sigma^*$, $xz$ is in $L$ exactly when $yz$ is in $L$. Then $R_L$ is of finite index [4].

The remainder of this paper demonstrates an application of the pumping lemma to prove that an even-length palindrome is a non-regular language. It then shows why the pumping lemma cannot be used to similarly prove that an even-length palindrome followed by an arbitrary suffix is non-regular. It concludes by showing that such a palindrome-with-suffix language can still be proved non-regular through use of the Myhill-Nerode theorem.

## II. Even-Length Palindromes Are Non-Regular

Consider the following language: $L = \{rr^R | r \in \{\texttt{0},\texttt{1}\}^+\}$. This is the set of strings in which each string begins with a sequence of one or more `0`s or `1`s and then immediately concludes with an identical-but-reversed sequence. In other words, $L$ is the set of symmetric even-length palindromes over the alphabet $\{\texttt{0},\texttt{1}\}$. The pumping lemma can be used to show that this language is not regular.

Proof: Assume that $L$ were regular. Let $p$ be the length specified by the pumping lemma. Let $w$ be the string $\texttt{0}^p\texttt{11}\texttt{0}^p$. $w$ is a member of $L$ and $|w| \geq p$. Therefore, the pumping lemma states that $w$ can be subdivided, $w = xyz$, such that $|xy| \leq p$, $|y| \geq 1$, and $xy^iz \in L$ for all $i \geq 0$.

Since $|xy| \leq p$, any selection of $xy$ must fall within the $\texttt{0}^p$ prefix of $w$. This means $y$ consists of one or more `0`s. Thus, pumping the string $xy^iz$ for any $i > 1$ produces a string with more `0`s before the `11` portion of $w$ than after it. For example, if $i = 2$, the resulting string is $\texttt{0}^{p+|y|}\texttt{11}\texttt{0}^p$. This string is not in $L$, which violates our assumption that $L$ is regular.

This contradiction shows that $L$ is not regular.

## III. The Addition of a Suffix to an Even-Length Palindrome

Consider a slightly different language: $L_s = \{rr^Rs | r, s \in \{\texttt{0},\texttt{1}\}^+\}$.

This language makes for a trickier pumping lemma proof because the boundaries between the substrings $r$, $r^R$, and $s$ can be redrawn after pumping. As an example, consider the string $\texttt{0}^p\texttt{11}\texttt{0}^p\texttt{1}$ where $p$ is the length specified by the pumping lemma. An adversary may simply select $w = xyz$ such that $x = \epsilon$ and $y =$ the first character of the string (`0`). As in the proof above, pumping this for any positive $i$ results in more `0`s before the `11` portion $w$ than after it. However, the resulting string is still in $L_s$ if we consider it as follows: $r = \texttt{0}$, $r^R = \texttt{0}$, and $s$ is the remainder of the string.

In fact, any string of non-trivial length in $L_s$ can be pumped in a manner similar to this.

### A. Inadequacy of the Pumping Lemma

Since $r$ and $s$ must each contain at least one symbol, the shortest string in $L_s$ is 3 symbols in length. But all strings in $L_s$ of length 4 or longer can be pumped according to the pumping lemma, as follows.

Assume that both $|w|$ and the pumping length, $p$, are greater than 3. Also $r = a_1...a_n$, $r^R = a_n...a_1$, and $s = b_1...b_n$ where $a_i, b_i \in \{\texttt{0},\texttt{1}\}$ and $n \geq 1$.

Case 1: If $|r| > 1$, pump the first symbol in $r$. That is, the adversary can divide $w = rr^Rs = xyz$ such that $x = \epsilon, y = a_1$, and $z = a_2...a_na_n...a_1b_1...b_n$. If pumped negatively to produce $w' = xy^0z$, the resulting string $w'$ is still in the language, with $r' = a_2...a_n$, $r'^R = a_n...a_2$, and $s' = a_1b_1...b_n$. If pumped positively to produce $w' = xy^iz$ where $i \geq 2$, the resulting string $w'$ is still in the language, with $r' = a_1$, $r'^R = a_1$, and $z' = a_1^{i-2}a_2...a_na_n...a_1b_1...b_n$.

Case 2: If $|r| = 1$, pump the first symbol in $s$ instead. That is, if $|r| = 1$, then $|r^R| = 1$ as well. Since the initial string $w$ is of length 4 or greater, $|s| \geq 2$. The adversary can then divide $w = rr^R s = xyz$ such that $x = rr^R, y = b_1$, and $z = b_2...b_n$. Because $|rr^R| = 2$ and $p \geq 4$, this selection is always valid since $|xy| < p$, as the pumping lemma requires. Any pumping, whether positive or negative, leaves the $rr^R$ portion of the original string unchanged. If pumped negatively to produce $w' = xy^0z$, the original $s$ loses its first character ($b_1$), but still retains at least one character ($b_2$). If pumped positively so that $b_1$ is duplicated, the resulting $s$ portion of the string is longer but still valid.

Therefore, any string of length 4 or longer from $L_s$ can be pumped according to the rules of the pumping lemma.

As discussed previously, this does not prove $L_s$ to be regular. It simply means we cannot prove that $L_s$ is non-regular with this method.

### *B. Proof by Myhill-Nerode Theorem*

Consider the set of strings $S = \{(\texttt{10})^m\texttt{11}(\texttt{01})^n0 | m, n \geq 1 \text{ and } m \leq n\}$. Note that only those strings for which $m \leq n$ are in the language $L_s$. The sequence $(\texttt{10})^m$ cannot produce an internal even-length palindrome because it cannot produce any two identical symbols in a row, which is required to form any symmetrical reflection point between $r$ and $r^R$. Therefore, the form $rr^R$ requires that $m$ at least equal $n$. However, $n$ may exceed $m$ since any extra $(\texttt{01})$ sequences can be considered to be part of $s$.

So $S$ is a subset of $L_s$. Assume that $L_s$ is regular.

Let $x = (\texttt{10})^i$ and $y = (\texttt{10})^j$ such that $i$ and $j$ are any positive integers and $i \neq j$. Let $z = \texttt{11}(\texttt{01})^k0$ where $k$ is the lesser of $i$ and $j$. Observe that either $xz$ or $yz$ is in $L_s$, but not both, for this particular $z$.

Recall that the third statement of the Myhill-Nerode theorem states that, for a regular language: "Let equivalence relation $R_L$ be defined by: $xR_Ly$ if and only if for all $z$ in $\Sigma^*$, $xz$ is in $L$ exactly when $yz$ is in $L$. Then $R_L$ is of finite index" [4].

Here, for at least one $z$, $xz$ is in $L_s$ while $yz$ is not. This means $xz$ and $yz$ do not share an equivalence class, but each form their own. Since $i$ and $j$ can be any positive integers, the number of such equivalence classes is infinite for both $S$ and $L_s$. This contradicts the Myhill-Nerode theorem's statement that, in a regular language, such equivalences are always of finite index.

The assumption that $L_s$ is regular leads to this contradiction. Therefore, $L_s$ is not regular.

## IV. Conclusion

Using the pumping lemma, this paper has shown that an even-length palindrome is not regular. It has also shown that the addition of an arbitrary suffix to such a palindrome means that the pumping lemma is no longer adequate in proving the resulting language non-regular. Instead, the Myhill-Nerode theorem must be used.

It is left to future work to prove the non-regularity of the broader classes of palindrome-based languages, including those that are not of even length, include an arbitrary prefix, or include both prefix and suffix.

## References


[1] K. Sugihara. (2011, Apr.) ICS241 Lecture Notes #20. [Online]. Available: http://pearl.ics.hawaii.edu/ sugihara/course/ics241s11/notes/04-05n20.html

[2] Regular Language. (2012, Feb.) [Online]. http://en.wikipedia.org/w/index.php?title=Regular_language&oldid=475373506

[3] M. Sipser. "Regular Languages", in *Introduction to the Theory of Computation*, 1st ed. Boston, MA: PWS Publishing Company, 1997, ch. 1.

[4] J. E. Hopcroft and J. D. Ullman. "Properties of Regular Sets", in *Introduction to Automata Theory, Languages, and Computation*, 1st ed. Addison-Wesley Publishing Company, 1979, ch. 3.

[5] M. Fleck and M. Parthasarathy. (2006, Fall) Pumping Lemma Help. [Online]. Available: http://www.cs.uiuc.edu/class/fa06/cs273/Lectures/pumping-lemma/

[6] S. Toida. (2011, Jan.) Non-Regular Languages. [Online.] Available: http://www.cs.odu.edu/ toida/nerzic/390teched/regular/reg-lang/non-regularity.html